CNC USB Controller API

User manual

2010-09-20

Table of Contents

1 API Reference....4
1.1 Introduction....4
1.2 General functions....5
1.2.1 About....5
1.3 HID functions....6
1.3.1 HidInitialize....6
1.3.2 HidFinalize....6
1.3.3 CheckHidInitialized....7
1.3.4 GetHidDeviceList....7
1.3.5 GetHidDeviceInfo....8
1.3.6 GetHidDeviceInfoByName....8
1.4 Controller functions....9
1.4.1 Connect....9
1.4.2 Disconnect....9
1.4.3 IsConnected....9
1.4.4 GetDeviceInfo....10
1.4.5 GetDeviceInfoEx....11
1.4.6 RegisterStatusCallback....12
1.4.7 RegisterDataCallback....12
1.4.8 GetState....13
1.5 Controller command functions....14
1.5.1 SetDeviceParams....14
1.5.2 CmdInfo....15
1.5.3 CmdWrite....15
1.5.4 CmdMoveAxes....16
1.5.5 CmdMoveAxesDelta....17
1.5.6 CmdPos....18
1.5.7 CmdDwell....18
1.5.8 CmdEStop....19
1.5.9 CmdStop....19
1.5.10 CmdPause, CmdPauseNow....20
1.5.11 CmdResume, CmdResumeNow....20
1.5.12 CmdOut, CmdOutNow....21
1.5.13 CmdSpindle, CmdSpindleNow....22
1.5.14 CmdCoolant, CmdCoolantNow....23
1.5.15 CmdNothing....24
1.6 Codes, constants, callbacks, ….....25
1.6.1 Error codes....25

1.6.2 Status codes....25
1.6.3 StatusCallback....26
1.6.4 DataCallback....27
1.6.5 CNC_CODE....28
1.6.6 PLANE....28
1.6.7 SPINDLE....28
1.6.8 COOLANT....28
2 Examples....29
2.1 About....29
2.2 HID information....29
2.3 Working with controller....29

1 API Reference

1.1 Introduction

All functions are exported using stdcall calling convention.
Following attributes are used for all described functions

C#

```
const string CNCUSB_DLL = "cncusb.dll";

[DllImport(
      CNCUSB_DLL,
      SetLastError = false,
      ExactSpelling = true,
      CharSet = CharSet.Unicode,
      CallingConvention = CallingConvention.StdCall)]
public static extern ...
```

Delphi

```
const CNCUSB_DLL = 'cncusb.dll';

function ... stdcall; external 'cncusb.dll';
```

Be careful when using boolean in Delphi. Delphi types Boolean, BOOL and LongBool are NOT the same. You must always use BOOL with this API. Also use WideString instead of string.

1.2 General functions

1.2.1 About

Function will return information about this API.

Function declarations (calling convention attributes are not displayed)

C#
`ERROR About([MarshalAs(UnmanagedType.BStr)] out string msg);`
Delphi
`function About(out msg: WideString): ERROR;`

Function parameters

Name	Type	Description
`Return value`	`ERROR`	`ERROR code`
`msg`	`[out] string`	`Information about this API`

1.3 HID functions

1.3.1 HidInitialize

Function will initialize HID subsystem.

Function declarations (calling convention attributes are not displayed)

C#
`ERROR HidInitialize();`
Delphi
`function HidInitialize(): ERROR;`

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code

1.3.2 HidFinalize

Function will close HID subsystem.

Function declarations (calling convention attributes are not displayed)

C#
`ERROR HidFinalize();`
Delphi
`function HidFinalize(): ERROR;`

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code

1.3.3 CheckHidInitialized

Function will check, if HID subsystem is initialized.

Function declarations (calling convention attributes are not displayed)

C#

```
[return: MarshalAs(UnmanagedType.Bool)]
bool CheckHidInitialized();
```

Delphi

```
function CheckHidInitialized(): BOOL;
```

Function parameters

Name	Type	Description
Return value	bool	True if HID is initialized

1.3.4 GetHidDeviceList

Function will return list of all HID devices.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR GetHidDeviceList(
     [MarshalAs(UnmanagedType.BStr)] out string deviceList);
```

Delphi

```
function GetHidDeviceList(out deviceList: WideString): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
deviceList	[out] string	List of all HID devices

1.3.5 GetHidDeviceInfo

Function will return device vendor, product name and serial number.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR GetHidDeviceInfo(
     int vid, int pid,
     [MarshalAs(UnmanagedType.BStr)] out string vendorName,
     [MarshalAs(UnmanagedType.BStr)] out string productName,
     [MarshalAs(UnmanagedType.BStr)] out string serialNumber);
```

Delphi

```
function GetHidDeviceInfo(
     vid, pid: integer;
     out vendorName, productName, serialNumber: WideString): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code

1.3.6 GetHidDeviceInfoByName

Function will return device vendor, product name and serial number.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR GetHidDeviceInfoByName(
     [MarshalAs(UnmanagedType.BStr)] string device,
     [MarshalAs(UnmanagedType.BStr)] out string vendorName,
     [MarshalAs(UnmanagedType.BStr)] out string productName,
     [MarshalAs(UnmanagedType.BStr)] out string serialNumber);
```

Delphi

```
function GetHidDeviceInfoByName(
     device: WideString;
     out vendorName, productName, serialNumber: WideString): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code

1.4 Controller functions

1.4.1 Connect

Function will initialize HID subsystem and connect API to controller.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR Connect();
```

Delphi

```
function Connect(): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code

1.4.2 Disconnect

Function will disconnect API from controller and close HID subsystem.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR Disconnect();
```

Delphi

```
function Disconnect(): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code

1.4.3 IsConnected

Function will check if API is connected to controller.

Function declarations (calling convention attributes are not displayed)

C#

```
[return: MarshalAs(UnmanagedType.Bool)]
bool IsConnected();
```

Delphi

```
function IsConnected(): BOOL;
```

Function parameters

Name	Type	Description
Return value	bool	True if controller is ready

1.4.4 GetDeviceInfo

Function will return vendor and product name of controller.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR GetDeviceInfo(
     [MarshalAs(UnmanagedType.BStr)] out string vendorName,
     [MarshalAs(UnmanagedType.BStr)] out string productName,
     [MarshalAs(UnmanagedType.Bool)]out bool diy);
```

Delphi

```
function GetDeviceInfo(
     out vendorName, productName: WideString; out diy: BOOL): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
vendorName	[out] string	www.planet-cnc.com
productName	[out] string	CNC USB Controller
diy	bool	True if controller id DIY version. DIY version is limited to 25 moves.

1.4.5 GetDeviceInfoEx

Function will return vendor and product name of controller.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR GetDeviceInfoEx(
    [MarshalAs(UnmanagedType.BStr)] out string vendorName,
    [MarshalAs(UnmanagedType.BStr)] out string productName,
    [MarshalAs(UnmanagedType.Bool)] out bool diy,
    out int serialNumber, out byte ver_year, out byte ver_month, out byte ver_day);
```

Delphi

```
function GetDeviceInfoEx(
    out vendorName, productName: WideString; out diy: BOOL;
    out serialNumber: integer; out ver_year, ver_month, ver_day: byte): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
vendorName	[out] string	www.planet-cnc.com
productName	[out] string	CNC USB Controller
diy	[out] bool	True if controller id DIY version. DIY version is limited to 25 moves.
serialNumber	[out] int	Controller serial number
ver_year	[out] byte	Controller version
ver_month	[out] byte	Controller version
ver_day	[out] byte	Controller version

1.4.6 RegisterStatusCallback

Function will register status callback.
Status callback will trigger when USB state is changed.

Function declarations (calling convention attributes are not displayed)

C#
`ERROR RegisterStatusCallback(StatusCallback statusCallback);`
Delphi
`function RegisterStatusCallback(statusCallback : TStatusCallback): ERROR;`

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
statusCallback	StatusCallback	Callback for receiving HID status events

1.4.7 RegisterDataCallback

Function will register data callback.
Data callback will trigger when API receives data from controller.

Function declarations (calling convention attributes are not displayed)

C#
`ERROR RegisterDataCallback(DataCallback dataCallback);`
Delphi
`function RegisterDataCallback(dataCallback : TDataCallback): ERROR;`

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
dataCallback	DataCallback	Callback for receiving controller events

1.4.8 GetState

Function will return last state received from controller.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR GetState(
      out CNC_CODE code,
      out byte bufferFree, out int numExecuting,
      out byte keysLimit, out byte keysJog,
      out int posA, out int posB, out int posC, out int posD,
      out byte output);
```

Delphi

```
function GetState(
        out code: CNC_CODE;
        out bufferFree: byte; out numExecuting: integer;
        out keysLimit, keysJog: byte;
        out posA, posB, posC, posD: integer;
        out output: byte): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
code	[out] CNC_CODE	status code
bufferFree	[out] byte	amount of free buffer
numExecuting	[out] int	number of executing command
keysLimit,keysJog	[out] byte	state of limit and jog keys
posA,posB,posC,posD	[out] int	position of axis in steps
output	[out] byte	state of controller output

1.5 Controller command functions

Commands with names starting with “Cmd” are actual commands send to controller.
All command functions returns number “num” which uniquely identifies command.
On DataCallback callback you can identify, which command is currently executing.

1.5.1 SetDeviceParams

Function will set machine parameters. You must call this before any other controller command functions.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR SetDeviceParams(
    byte microstepping,
    int axis1StepsPerMM, int axis2StepsPerMM,
    int axis3StepsPerMM, int axis4StepsPerMM,
    int axisLimitSwitch,
    double axis1Backlash, double axis2Backlash,
    double axis3Backlash, double axis4Backlash,
    double axis1Accel, double axis2Accel,
    double axis3Accel, double axis4Accel,
    double axis1InitSpeed, double axis2InitSpeed,
    double axis3InitSpeed, double axis4InitSpeed);
```

Delphi

```
function SetDeviceParams(
   axis1StepsPerMM, axis2StepsPerMM, axis3StepsPerMM, axis4StepsPerMM : integer;
   axisLimitSwitch : integer;
   axis1Backlash, axis2Backlash, axis3Backlash, axis4Backlash : double;
   axis1Accel, axis2Accel, axis3Accel, axis4Accel : double;
   axis1InitSpeed, axis2InitSpeed, axis3InitSpeed, axis4InitSpeed : double): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
axisStepsPerMM	int	Number of steps per millimeter
axisLimitSwitch	int	Bit encoded limit switch existence
axisBacklash	double	Backlash in millimeters
axisAccel	double	Acceleration in mm^2/sec (25 is default)
axisInitSpeed	double	Speed in mm/sec

1.5.2 CmdInfo

Function will force controller to send us new state information.

Function declarations (calling convention attributes are not displayed)

C#
`ERROR CmdInfo(out int num);`
Delphi
`function CmdInfo(out num: integer): ERROR;`

Function parameters

Name	Type	Description
`Return value`	`ERROR`	`ERROR code`
`num`	`[out] int`	`Command identifier`

1.5.3 CmdWrite

Function will force controller to internally store position and settings.

Function declarations (calling convention attributes are not displayed)

C#
`ERROR CmdInfo(out int num);`
Delphi
`function CmdInfo(out num: integer): ERROR;`

Function parameters

Name	Type	Description
`Return value`	`ERROR`	`ERROR code`
`num`	`[out] int`	`Command identifier`

1.5.4 CmdMoveAxes

Function will move axes to specified absolute position.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR CmdMoveAxes(out int num,
       double posA, double posB, double posC, double posD,
       double speed, bool startAccel, bool endAccel);
```

Delphi

```
function CmdMoveAxes(out num: integer;
       posA, posB, posC, posD: double;
       speed: double; startAccel, endAccel : BOOL): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
num	[out] int	Command identifier
posA, posB, posC, posD	double	Target position in millimeters
speed	double	Speed in millimeters per second
startAccel	bool	True if we want to accelerate at start
endAccel	bool	True if we want to decelerate at end

1.5.5 CmdMoveAxesDelta

Function will move axes to specified relative position.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR CmdMoveAxesDelta(out int num,
       double posDeltaA, double posDeltaB, double posDeltaC, double posDeltaD,
       double speed, bool startAccel, bool endAccel);
```

Delphi

```
function CmdMoveAxesDelta(out num: integer;
       posDeltaA, posDeltaB, posDeltaC, posDeltaD: double;
       double: speed; startAccel, endAccel : BOOL): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
num	[out] int	Command identifier
posDeltaA, posDeltaB, posDeltaC, posDeltaD	double	Delta position in millimeters
speed	double	Speed in millimeters per second
startAccel	bool	True if we want to accelerate at start
endAccel	bool	True if we want to decelerate at end

1.5.6 CmdPos

Function will set position of axes without axes movement.

Function declarations (calling convention attributes are not displayed)

C#
`ERROR CmdPos(out int num, double posA, double posB, double posC, double posD);`
Delphi
`function CmdPos(out num: integer; posA, posB, posC, posD: double): ERROR;`

Function parameters

Name	Type	Description
`Return value`	`ERROR`	`ERROR code`
`num`	`[out] int`	`Command identifier`
`posA, posB, posC, posD`	`double`	`Position in millimeters`

1.5.7 CmdDwell

Function will delay controller.

Function declarations (calling convention attributes are not displayed)

C#
`ERROR CmdDwell(out int num, double seconds);`
Delphi
`function CmdDwell(out num: integer; seconds: double): ERROR;`

Function parameters

Name	Type	Description
`Return value`	`ERROR`	`ERROR code`
`num`	`[out] int`	`Command identifier`
`seconds`	`double`	`Delay duration in seconds`

1.5.8 CmdEStop

Function will stop execution, clear outputs and clear all buffer. On DIY controllers it will alse reset move counter.

Function declarations (calling convention attributes are not displayed)

C#
`ERROR CmdEStop(out int num);`
Delphi
`function CmdEStop(out num: integer): ERROR;`

Function parameters

Name	Type	Description
`Return value`	`ERROR`	`ERROR code`
`num`	`[out] int`	`Command identifier`

1.5.9 CmdStop

Function will stop execution and clear all buffer.

Function declarations (calling convention attributes are not displayed)

C#
`ERROR CmdStop(out int num);`
Delphi
`function CmdStop(out num: integer): ERROR;`

Function parameters

Name	Type	Description
`Return value`	`ERROR`	`ERROR code`
`num`	`[out] int`	`Command identifier`

1.5.10 CmdPause, CmdPauseNow

Function will pause the controller.
Command with “Now” suffix will not be buffered and will execute immediately.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR CmdPause(out int num);
```

Delphi

```
function CmdPause(out num: integer): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
num	[out] int	Command identifier

1.5.11 CmdResume, CmdResumeNow

Function will resume paused controller.
Command with “Now” suffix will not be buffered and will execute immediately.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR CmdResume(out int num);
```

Delphi

```
function CmdResume(out num: integer): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
num	[out] int	Command identifier

1.5.12 CmdOut, CmdOutNow

Function will turn output pins on or off.
Command with “Now” suffix will not be buffered and will execute immediately.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR CmdOut(out int num, byte output);
ERROR CmdOutNow(out int num, byte output);
```

Delphi

```
function CmdOut(out num: integer; output: byte): ERROR;
function CmdOutNow(out num: integer; output: byte): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
num	[out] int	Command identifier
output	byte	Value of output pins

1.5.13 CmdSpindle, CmdSpindleNow

Function will set spindle. Output pins will turn on/off.
Command with “Now” suffix will not be buffered and will execute immediately.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR CmdSpindle(out int num, SPINDLE direction, double speed);
ERROR CmdSpindleNow(out int num, SPINDLE direction, double speed);
```

Delphi

```
function CmdSpindle(out num: integer; direction: SPINDLE; speed: double): ERROR;
function CmdSpindleNow(out num: integer; direction: SPINDLE; speed: double): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
num	[out] int	Command identifier
direction	SPINDLE	Spindle direction output
speed	double	Spindle speed (not implemented)

1.5.14 CmdCoolant, CmdCoolantNow

Function will set flood and mist coolant. Output pins will turn on/off.
Command with “Now” suffix will not be buffered and will execute immediately.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR CmdCoolant(out int num, COOLANT flood, COOLANT mist);
ERROR CmdCoolantNow(out int num, COOLANT flood, COOLANT mist);
```

Delphi

```
function CmdCoolant(out num: integer; flood, mist: COOLANT): ERROR;
function CmdCoolantNow(out num: integer; flood, mist: COOLANT): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
num	[out] int	Command identifier
flood	COOLANT	Flood output
mist	COOLANT	Mist output

1.5.15 CmdNothing

Function will do nothing.

Function declarations (calling convention attributes are not displayed)

C#

```
ERROR CmdNothing(out int num);
```

Delphi

```
function CmdNothing(out num: integer): ERROR;
```

Function parameters

Name	Type	Description
Return value	ERROR	ERROR code
num	[out] int	Command identifier

1.6 Codes, constants, callbacks, …

1.6.1 Error codes

Type: int

Name	Value	Description
SUCCESS	0x00000000	command completed successfully
FAIL	0x00000001	command failed
NOT_IMPLEMENTED	0x00000002	command is not implemented
NOT_INITIALIZED	0x00000003	HID is not initialized
DEVICE_NOT_FOUND	0x00000004	controller is not found
DEVICE_NOT_READY	0x00000005	controller is not ready
DEVICE_NOT_CONNECTED	0x00000006	controller is not connected
SEND_FAILED	0x00000007	send failed
ERR_LICENSE	0x00000008	license is invalid
ERR_RAWDATA	0x00000009	raw data is invalid
ERR_DEVICE_PARAMS	0x0000000A	device parameters are not correct

1.6.2 Status codes

Type: int

Name	Value	Description
NONE	0x00010000	nothing happened
CONNECTED	0x00010001	controller has just connected to USB
DISCONNECTED	0x00010002	controller has just disconnected from USB
UNPLUGED	0x00010003	controller is unplugged - it will be disconnected
DATAERROR	0x00010004	error in data transmission

1.6.3 StatusCallback

Callback for changes in UBS state

Function declarations

C#
`public delegate void StatusCallback(STATUS status);`
Delphi
`type TStatusCallback = procedure(status : STATUS); stdcall;`

Function parameters

Name	Type	Description
`status`	`STATUS`	`STATUS code`

1.6.4 DataCallback

Callback for changes in controller state

Function declarations

C#

```
public delegate void DataCallback(
              CNC_CODE code,
              byte bufferFree,
              int numExecuting,
              byte keysLimit, byte keysJog,
              int posA, int posB, int posC, int posD,
              byte output);
```

Delphi

```
type TDataCallback = procedure(
              code: CNC_CODE;
              bufferFree: byte;
              numExecuting: integer;
              keysLimit, keysJog: byte;
              posA, posB, posC, posD: integer;
              output: byte); stdcall;
```

Function parameters

Name	Type	Description
code	CNC_CODE	status code
bufferFree	byte	amount of free buffer
numExecuting	int	number of executing command
keysLimit	byte	state of limit keys
keysJog	byte	state of jog keys
posA	int	position of axis in steps
posB	int	position of axis in steps
posC	int	position of axis in steps
posD	int	position of axis in steps
output	byte	state of controller output

1.6.5 CNC_CODE

Type: byte

Name	Value	Description
Unknown	0	
OK	1	general info - everything is OK
Pos	2	general info - everything is OK
Stop	3	Machine cleared buffer and stopped
Error	8	error - command is ignored

1.6.6 PLANE

Type: byte

Name	Value	Description
Unknown	0	
XY	1	plane XY
YZ	2	plane YZ
ZX	3	plane ZX
UV	4	plane UV
VW	5	plane VW
UW	6	plane UW

1.6.7 SPINDLE

Type: byte

Name	Value	Description
Unknown	0	
Stopped	1	stopped
CW	2	rotating CW
CCW	3	rotating CCW

1.6.8 COOLANT

Type: byte

Name	Value	Description
Unknown	0	
Off	1	coolant is off
On	2	coolant is on

2 Examples

2.1 CNCUSBControllerAPI_demo

CNCUSBControllerAPI_demo is example application written in C#.